CLAIMS

What is claimed is:

1. A system, comprising:

an escape analysis module to (1) determine which objects of a program can be stack allocated under a closed-world assumption and (2) analyze which stack allocation is invalidated due to an occurrence of an open-world feature;

a stack allocation module to stack allocate the objects determined by the escape analysis module; and

a stack allocation recovery module to recover invalidated stack allocations back to their original allocations in heap based on the analysis of the escape analysis module.

2. The system of claim 1, further comprising a main engine to, when the open-world feature occurs, invoke (1) the escape analysis module to check which stack allocation is invalidated and to identify allocation sites of the invalidated stack allocations, and (2) the stack allocation recovery module to recover the invalidated stack allocations.

3. The system of claim 2, wherein the main engine, the escape analysis module, the stack allocation module, and the stack allocation recovery module are all part of a runtime environment.

4. The system of claim 1, wherein the stack allocation recovery module recovers the invalidated stack allocations back to their original allocations in the heap by:

suspending all threads so that safe patching and compensation publishing can be made;

patching each allocation site of the invalidated stack allocations back to its original allocation in the heap; and

performing compensation publishing where necessary.

5. The system of claim 4, wherein the stack allocation recovery module performs the patching by patching stack allocation instructions back to heap allocation instructions.

6. The system of claim 4, wherein the stack allocation recovery module performs the compensation publishing by:

enumerating all stack object references to objects allocated on a stack into a record set;

going through all stack frames and identifying allocation sites of the invalidated stack allocations within each frame;

in each invalidated allocation site, checking if there is any stack object allocated at this site by searching the record set, wherein objects found in the record set are added, as initial publishing candidates, into a compensation publication;

publishing the compensation publication to the heap beginning with the initial publishing candidates.

7. A method of permitting stack allocation in a program with open-world features, comprising:

determining which objects of the program can be stack-allocated under a closed-world assumption;

stack allocating these objects based on the determination;

analyzing which stack allocation is invalidated due to the occurrence of an open-world feature;

recovering those invalidated stack allocations back to their original allocations in heap based on the analysis.

8. The method of claim 7, wherein recovering those invalidated stack allocations back to their original allocations in the heap comprises:

suspending all threads so that safe patching and compensation publishing can be made;

patching each allocation site of the invalidated stack allocations back to its original allocation in the heap; and

performing compensation publishing where necessary.

9. The method of claim 8, wherein patching each allocation site of the invalidated stack allocations comprises patching stack allocation instructions back to heap allocation instructions.

10. The method of claim 8, wherein compensation publishing comprises:

enumerating all stack object references to objects allocated on a stack into a record set;

going through all stack frames and identifying allocation sites of the invalidated stack allocations within each frame;

in each invalidated allocation site, checking if there is any stack object allocated at this site by searching the record set, wherein the objects found in the record set are added, as initial publishing candidates, into a compensation publication; and

publishing the compensation publication to the heap beginning with the initial publishing candidates.

11. The method of claim 7, wherein (1) determining which objects of the program can be stack-allocated, (2) stack allocating the objects that can be stack allocated, (3) analyzing which stack allocation is invalidated, and (4) recovering those invalidated stack allocations are all performed within a runtime environment.

12. The method of claim 7, wherein analyzing which stack allocation is invalidated is performed when the open-world feature occurs.

13. The method of claim 7, wherein recovering those invalidated stack allocations is performed after analyzing which stack allocation is invalidated.

14. A machine-readable medium having stored thereon sequences of instructions, the sequences of instructions including instructions which, when executed by a processor, causes the processor to perform:

determining which objects of a program can be stack-allocated under a closed-world assumption;

stack allocating the objects based on the determination;

analyzing which stack allocation is invalidated due to an occurrence of an open-world feature;

recovering invalidated stack allocations back to their original allocations in heap based on the analysis.

15. The machine-readable medium of Claim 14, wherein recovering invalidated stack allocations comprises:

suspending all threads so that safe patching and compensation publishing can be made;

patching each allocation site of the invalidated stack allocations back to its original allocation in the heap; and

performing compensation publishing where necessary.

16. The machine-readable medium of Claim 15, wherein patching each allocation site of the invalidated stack allocations comprises patching stack allocation instructions back to heap allocation instructions.

17. The machine-readable medium of Claim 15, wherein performing the compensation publishing comprises:

enumerating all stack object references to objects allocated on a stack into a record set;

going through all stack frames and identifying allocation sites of the invalidated stack allocations within each frame;

in each invalidated allocation site, checking if there is any stack object allocated at this site by searching the record set, wherein the objects found in the record set are added, as initial publishing candidates, into a compensation publication; and

publishing the compensation publication to the heap beginning with the initial publishing candidates.

18. The machine-readable medium of Claim 14, wherein analyzing which stack allocation is invalidated is performed when the open-world feature occurs.

19. The machine-readable medium of Claim 14, wherein recovering those invalidated stack allocations is performed after analyzing which stack allocation is invalidated.

20. The machine-readable medium of Claim 14, wherein the instructions are part of a runtime environment.

21. The machine-readable medium of Claim 14, wherein the machine-readable medium is a memory within a computer system.